

CHAPTER 5



MongoDB - Installation and Configuration

*“MongoDB is a **cross-platform database**.”*

In this chapter, you will go over the process of installing MongoDB on Windows and Linux.

Select Your Version

MongoDB runs on most platforms. A list of all the available packages is available on the MongoDB downloads page at www.mongodb.org/downloads.

The correct version for your environment will depend on your server's operating system and the kind of processor. MongoDB supports both 32-bit and 64-bit architecture but it's recommended to use 64-bit in your production environment.

■ **32-bit limitation** This is due to the usage of memory mapped files in MongoDB. This limits the **32-bit** builds to around 2GB of data. It's recommended to use a **64-bit** build for a production environment for performance reasons.

The latest MongoDB production release is **3.0.4** at the time of writing this book. Downloads for MongoDB are available for **Linux, Windows, Solaris, and Mac OS X**.

The MongoDB download page is divided in the following sections:

- Current **Stable** Release (3.0.4) – 6/16/2015
- Previous Releases (stable)
- Development Releases (unstable)

The current release is the most stable recent version available, which at time of writing of the book is 3.0.4. When a new version is released, the prior stable release is moved to the Previous Releases section.

The development releases, as the name suggests, are the versions that are still under development and hence are tagged as unstable. These versions can have additional features but they may not be stable since they are still in the development phase. You can use the development versions to try out new features and provide feedback to 10gen regarding the features and issues faced.

Installing MongoDB on Linux

This section covers installing MongoDB on a LINUX system. For the following demonstration, we will be using an Ubuntu Linux distribution. You can install MongoDB either manually or via repositories. We will walk you through both options.

Installing Using Repositories

In LINUX, repositories are the online directories that contain software. Aptitude is the program used to install software on Ubuntu. Although MongoDB might be present in the default repositories, there is the possibility of an out-of-date version, so the first step is to configure Aptitude to look at the custom repository.

1. Issue the following to import the `public.GPG` key for MongoDB:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
```

2. Next, use the following command to create the `/etc/apt/sources.list.d/mongodb-org-3.0.list` file:

```
echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb_release -sc)"/  
mongodb-org/3.0 multiverse" | sudo tee /etc/apt/sources.list.d/  
mongodb-org-3.0.list
```

3. Finally, use the following command to reload the repository:

```
sudo apt-get update
```

Now Aptitude is aware of the manually added repository.

4. Next, you need to install the software. The following command should be issued in the shell to install MongoDB's current stable version:

```
sudo apt-get install -y mongodb-org
```

You've successfully installed MongoDB, and that's all there is to it.

Installing Manually

In this section, you will see how MongoDB can be installed manually. This knowledge is important in the following cases:

- When the Linux distribution doesn't use Aptitude.
- When the version you require is not available through repositories or is not part of the repository.
- When you need to run multiple MongoDB versions simultaneously.

The first step in a manual installation is to decide on the version of MongoDB to use and then download from the site. Next, the package needs to be extracted using the following command:

```
$ tar -xvf mongodb-linux-x86_64-3.0.4.tgz
mongodb-linux-i686-3.0.4/THIRD-PARTY-NOTICES
mongodb-linux-i686-3.0.4/GNU-AGPL-3.0
mongodb-linux-i686-3.0.4/bin/mongodump
.....
mongodb-linux-i686-3.0.4/bin/mongosniff
mongodb-linux-i686-3.0.4/bin/mongod
mongodb-linux-i686-3.0.4/bin/mongos
mongodb-linux-i686-3.0.4/bin/mongo
```

This extracts the package content to a new directory, namely `mongodb-linux-x86_64-3.0.4` (which is located under your current directory). The directory contains many subdirectories and files. The main executable files are under the subdirectory `bin`.

This completes the MongoDB installation successfully.

Installing MongoDB on Windows

Installing MongoDB on Windows is a simple matter of downloading the msi file for the selected build of Windows and running the installer.

The **installer** will guide you through installation of MongoDB.

Following the wizard, you will reach the Choose Setup Type screen. There are two setup types available wherein you can customize your installation. In this example, select the setup type as Custom.

An installation directory needs to be specified when selecting Custom, so specify the **directory** to **C:\PracticalMongoDB**.

Note that MongoDB can be run from any folder selected by the user because it is self-contained and has no dependency on the system. If the setup type of Complete is selected, the default folder selected is `C:\Program Files\MongoDB`.

Clicking Next will take you to the Ready to installation screen. Click Install.

This will start the installation and will show the progress on a screen. Once the installation is complete, the wizard will take you to the completion screen.

Clicking Finish completes the setup. After successful completion of the above steps, you have a directory called `C:\PracticalMongoDB` with all the relevant applications in the **bin** folder. That's all there is to it.

Running MongoDB

Let's see how to start running and using MongoDB.

Preconditions

A data folder is required for storing the files. This by default is **C:\data\db** in Windows and `/data/db` in LINUX systems.

These data directories are not created by MongoDB, so before starting MongoDB the data directory **needs to be manually created** and you need to ensure that **proper permissions are set** (such as that MongoDB has read, write, and directory creation permissions).

If the MongoDB is started before you create the folder, it will **throw an error message** and will fail to run.

Starting the Service

Once the directories are created and permissions are in place, execute the **mongod** application (placed under the **bin** directory) to start the **MongoDB core database service**.

In continuation of the above installation, the same can be started by opening the command prompt in Windows (which needs to be run as administrator) and executing the following:

```
c:\>c:\practicalmongodb\bin\mongod.exe
```

In case of Linux, the **mongod** process is started in the shell.

This will start the MongoDB database on the localhost interface. It will listen for connections from the mongo shell on port 27017.

As mentioned, the folder path needs to be created before starting the database, which by default is **c:\data\db**. An alternative path can also be provided when starting the database service by using the **-dbpath** parameter.

```
C:\>C:\practicalmongodb\bin\mongod.exe --dbpath
C:\NewDBPath\DBContents
```

Verifying the Installation

The relevant executable will be present under the subdirectory **bin**. The following can be checked under the **bin** directory in order to vet the success of the installation step:

- **Mongod**: the core database server
- **Mongo**: The database shell
- **Mongos**: The auto-sharding process
- **Mongoexport**: The export utility
- **Mongoimport**: The import utility

Apart from the above, there are other applications available in the **bin** folder.

The **mongo** application launches the **mongo shell**, which supplies access to the database contents and lets you fire selective queries or execute aggregation against the data in MongoDB.

The **mongod** application, as you saw above, is used to **start the database service**, or daemon.

Multiple flags can be set when launching the applications. For example, **-dbpath** can be used to specify an alternative path for where the database files should be stored. To get the list of all available options, include the **--help** flag when launching the service.

MongoDB Shell

The **mongo** shell comes as part of the standard distribution of MongoDB. The shell provides a **full database interface** for MongoDB, enabling you play around with the data stored in MongoDB using a **JavaScript environment**, which has complete access to the **language** and all the **standard functions**.

Once database services have started, you can fire up the **mongo** shell and start using MongoDB. This can be done using Shell in Linux or the command prompt in Windows (**run as administrator**).

You must refer to the exact location of the executable, such as in the **C:\practicalmongodb\bin** folder in a Windows environment.

Open the command prompt (run as administrator) and type `mongo.exe`. Press the Enter key. This will start the mongo shell.

```
C:\>C:\practicalmongodb\bin\mongo.exe
MongoDB shell version: 3.0.4
connecting to: test
```

>

If no parameters are specified when starting the service, it connects to the **default database** named `test` on the localhost instance.

The database will be created **automatically** when connected to it. MongoDB offers this feature of automatically creating a database if an attempt is made to access a one **that is not there**.

The next chapter offers more information on working with the mongo shell.

Securing the Deployment

You know how to install and start using **MongoDB** via the default configurations. Next, you need to ensure that the data that is stored within the database is **secure in all aspects**.

In this section, you will look at how to secure your data. You will change the configuration of the default installation to ensure that your database is more secure.

Using Authentication and Authorization

Authentication **verifies the user's identity**, and authorization determines the **level of actions** that the user can perform on the authenticated database.

This means the users will be able to access the database **only** if they log in using the **credentials** that have access on the database. This **disables anonymous access** to the database. After the user is authenticated, authorization can be used to ensure that the user has **only the required amount of access** needed to accomplish the tasks at hand.

Both authentication and authorization exist at a **per-database level**. The users exist in the context of a **single logical database**.

The information on the users is maintained in a collection named **system.users**, which exists in the `admin` database. This collection maintains the credentials needed for authenticating the user wherein it stores the user id, password, and the database against which it is created, plus privileges needed for authorizing the user.

MongoDB uses a role-based approach for authorization (the roles of `read`, `readWrite`, `readAnyDatabase`, etc.). If needed, the user administrator can **create custom roles**.

A privilege document within the `system.users` collection is used for storing each user roles. The same document maintains the credentials for authenticated users.

An example of a **document** in the `system.users` collection is as follows:

```
{
  _id : "practicaldb.Shaks",
  user : "Shaks",
  db : "practicaldb",
  credentials : {.....},
  roles : [
    { role: "read", db: "practicaldb" },
    { role: "readWrite", db: "MyDB" }
  ],
  .....
}
```

This document tells us that the user Shaks is associated with database `practicaldb` and it has read roles in the `practicaldb` database and a `readWrite` role in the `MyDB` database. Note that a user name and the associated database uniquely identifies a user within MongoDB, so if you have two users with the same name, but they are associated with different databases, then they are considered as two unique users. Thus a user can have multiple roles with different authorization levels on different databases.

The available roles are

- **read:** This provides a read-only access of all the collections for the specified database.
- **readWrite:** This provides a read-write access to any collection within the specified database.
- **dbAdmin:** This enables the user to **perform administrative actions** within the **specified** database such as index management using `ensureIndex`, `dropIndexes`, `reIndex`, `indexStats`, renaming collections, creating collections, etc.
- **userAdmin:** This enables the user to perform **readWrite operations** on the **system.users** collection of the specified database. It also enables altering permissions of existing users or creating new users. This is effectively the `SuperUser` role for the specified database.
- **clusterAdmin:** This role enables the user to grant access to administrative operations that alter or display information about the **complete system**. `clusterAdmin` is applicable only on the `admin` database.
- **readAnyDatabase:** This role enables user to **read** from any database in the MongoDB environment.
- **readWriteAnyDatabase:** This role is similar to **readWrite** except it is for all databases.
- **userAdminAnyDatabase:** This role is similar to the **userAdmin** role except it applies to all databases.
- **dbAdminAnyDatabase:** This role is the same as **dbAdmin**, except it applies to all databases.
- Starting from version 2.6, a user admin can also create user-defined roles to adhere to the policy of least privilege by providing access at collection level and command level. A user-defined role is scoped to the database in which it's created and is uniquely identified by the combination of the database and the role name. All the user defined roles are stored in the `system.roles` collection.

Enabling Authentication

Authentication is **disabled by default**, so use **`--auth`** to enable **authentication**. While starting `mongod`, use `mongod --auth`. Before enabling authentication, you need to have at **least one admin user**. As you saw above, an admin user is a user who is responsible for **creating and managing other users**.

It is recommended that in production deployments such users are created solely for managing users and should not be used for any other roles. In a MongoDB deployment, this user is the first user that needs to be created; other users of the system can be created by this user.

The admin user can be created either way: before enabling the authentication or after enabling the authentication.

In this example, you will first **create the admin user and then enable the auth settings**. The below steps should be executed on the Windows platform.

Start the mongod with default settings:

```
C:\>C:\practicalmongodb\bin\mongod.exe
C:\practicalmongodb\bin\mongod.exe --help for help and startup options

2015-07-03T23:11:10.716-0700 I CONTROL   Hotfix KB2731284 or later update is installed, no
need to zero out data files
2015-07-03T23:11:10.716-0700 I JOURNAL   [initandlisten] journal dir=C:\data\db\journal
.....

2015-07-03T23:11:10.763-0700 I CONTROL   [initandlisten] MongoDB starting : pid=2776
port=27017 dbpath=C:\data\db\ 64-bit host=ANOC9
2015-07-03T23:11:10.763-0700 I CONTROL   [initandlisten] targetMinOS: Windows 7/W
indows Server 2008 R2
2015-07-03T23:11:10.763-0700 I CONTROL   [initandlisten] db version v3.0.4
2015-07-03T23:11:10.764-0700 I CONTROL   [initandlisten] OpenSSL version: OpenSSL
1.0.1j-fips 19 Mar 2015
2015-07-03T23:11:10.764-0700 I CONTROL   [initandlisten] build info: windows sys.
getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1')
BOOST_LIB_VERSION=1_49
2015-07-03T23:11:10.771-0700 I NETWORK   [initandlisten] waiting for connections
on port 27017
```

Creating the Admin User

Run another instance of command prompt by running it as an administrator and execute the mongo application:

```
C:\>C:\practicalmongodb\bin\mongo.exe
MongoDB shell version: 3.0.4
connecting to: test
>
```

Switching to the Admin Database

Note that admin db is a privileged database that the user needs access to in order to execute certain administrative commands such as creating an admin user.

```
>db = db.getSiblingDB('admin')
```

Admin

The user needs to be created with either of the roles: `userAdminAnyDatabase` or `userAdmin`:

```
>db.createUser({user: "AdminUser", pwd: "password", roles:["userAdminAnyDatabase"]})
Successfully added user: { "user" : "AdminUser", "roles" : [ "userAdminAnyDatabase" ] }
```

Next, authenticate using this user. **Restart the mongod with auth settings:**

```
C:\>C:\practicalmongodb\bin\mongod.exe -auth
C:\practicalmongodb\bin\mongod.exe --help for help and startup options
2015-07-03T23:11:10.716-0700 I CONTROL Hotfix KB2731284 or later update is installed, no
need to zero out data files
2015-07-03T23:11:10.716-0700 I JOURNAL [initandlisten] journal dir=C:\data\db\journal
.....
2015-07-03T23:11:10.763-0700 I CONTROL [initandlisten] MongoDB starting : pid=2776
port=27017 dbpath=C:\data\db\ 64-bit host=ANOC9
2015-07-03T23:11:10.763-0700 I CONTROL [initandlisten] targetMinOS: Windows 7/W
indows Server 2008 R2
2015-07-03T23:11:10.763-0700 I CONTROL [initandlisten] db version v3.0.4
2015-07-03T23:11:10.764-0700 I CONTROL [initandlisten] OpenSSL version: OpenSSL
1.0.1j-fips 19 Mar 2015
2015-07-03T23:11:10.764-0700 I CONTROL [initandlisten] build info: windows sys.
getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1')
BOOST_LIB_VERSION=1_49
2015-07-03T23:11:10.771-0700 I NETWORK [initandlisten] waiting for connections
on port 27017
```

Start the **mongo console** and authenticate against the admin database using the AdminUser user created above:

```
C:\>c:\practicalmongodb\bin\mongo.exe
MongoDB shell version: 3.0.4
connecting to: test
>use admin
switched to db admin
>db.auth("AdminUser", "password")
1
>
```

Creating a User and Enabling Authorization

In this section, you will **create a user and assign a role** to the newly created user. You have already authenticated using the admin user, as shown:

```
C:\>c:\practicalmongodb\bin\mongo.exe
MongoDB shell version: 3.0.4
connecting to: test
>use admin
switched to db admin
>db.auth("AdminUser", "password")
1
>
```


Switch to the Product database and create user Alice and assign read access on the product database, like so:

```
>use product
switched to db product
>db.createUser({user: "Alice"
... , pwd:"Moon1234"
... , roles: ["read"]
... }
... )
Successfully added user: { "user" : "Alice", "roles" : [ "read" ] }
```

Next, validate that the user has read-only access on the database:

```
>db
product
>show users
{
  "_id" : "product.Alice",
  "user" : "Alice",
  "db" : "product",
  "roles" : [
    {
      "role" : "read",
      "db" : "product"
    }
  ]
}
```

Next, connect to a **new mongo console** and log in as Alice to the Products database to issue read-only commands:

```
C:\>c:\practicalmongodb\bin\mongo.exe -u Alice -p Moon1234 product
2015-07-03T23:11:10.716-0700 I CONTROL Hotfix KB2731284 or later update is installed, no
need to zero-out data files
MongoDB shell version: 3.0.4
connecting to: products
```

Post successful authentication the following entry will be seen on the mongod console.

```
2015-07-03T23:11:26.742-0700 I ACCESS [conn2] Successfully authenticated as principal
Alice on product
```

Controlling Access to a Network

By default, mongod and mongos bind to all the available IP addresses on a system. In this section, you will look at configuration options for restricting network exposure. The code below is executed on the Windows platform:

```
C:\>c:\practicalmongodb\bin\mongod.exe --bind_ip 127.0.0.1 --port 27017 --rest
2015-07-03T00:33:49.929-0700 I CONTROL Hotfix KB2731284 or later update is installed, no
need to zero out data files
2015-07-03T00:33:49.946-0700 I JOURNAL [initandlisten] journal dir=C:\data\db\journal
2015-07-03T00:33:49.980-0700 I CONTROL [initandlisten] MongoDB starting : pid=1144
port=27017 dbpath=C:\data\db\ 64-bit host=ANOC9
2015-07-03T00:33:49.980-0700 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows
Server 2008 R2
2015-07-03T00:33:49.980-0700 I CONTROL [initandlisten] db version v3.0.4
2015-07-03T00:33:49.980-0700 I CONTROL [initandlisten] OpenSSL version: OpenSSL1.0.1j-fips
19 Mar 2015
2015-07-03T00:33:49.980-0700 I CONTROL [initandlisten] build info: windows
sys.getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1')
BOOST_LIB_VERSION=1_49
2015-07-03T00:33:49.981-0700 I CONTROL [initandlisten] allocator: system
2015-07-03T00:33:49.981-0700 I CONTROL [initandlisten] options: { net: { bindIp:
"127.0.0.1", http: { RESTInterfaceEnabled: true, enabled: true }, port: 27017} }
2015-07-03T00:33:49.990-0700 I NETWORK [initandlisten] waiting for connections on port
27017
2015-07-03T00:33:49.990-0700 I NETWORK [websvr] admin web console waiting for connections
on port 28017
2015-07-03T00:34:22.277-0700 I NETWORK [initandlisten] connection accepted from
127.0.0.1:49164 #1 (1 connection now open)
```

You have started the server with `bind_ip`, which has one value set as `127.0.0.1`, which is the localhost interface.

The `bind_ip` limits the network interfaces of the incoming connections for which the program will listen. Comma-separated IP addresses can be specified. In your case, you have restricted the mongod to listen to only the localhost interface.

When the mongod instance is started, by default it waits for any incoming connection on port 27017. You can change this using `-port`.

Just changing the port does not reduce the risk much. In order to completely secure the environment, you need to allow only trusted clients to connect to the port using firewall settings.

Changing this port also changes the HTTP status interface port, which by default is 28017. This port is available on a port that is `X+1000`, where `X` represents the connection port.

This web page exposes diagnostic and monitoring information, which includes operational data, a variety of logs, and status reports regarding the database instances. It provides management-level statistics that can be used for administration purpose. This page is by default read-only; to make it fully interactive, you will use the REST settings. This configuration makes the page fully interactive, helping the administrators troubleshoot any performance issues. Only trusted client access should be allowed on this port using firewalls.

It is recommended to disable the HTTP Status page as well as the REST configuration in the production environment.

Use Firewalls

Firewalls are used to control access within a network. They can be used to allow access from a specific IP address to specific IP ports, or to stop any access from **any untrusted hosts**. They can be used to create a trusted environment for your **mongod** instance where you can specify what IP addresses or hosts can connect to which ports or interfaces of the mongod.

On the Windows platform, use netsh to configure the incoming traffic for port 27017:

```
C:\>netsh advfirewall firewall add rule name="Open mongod port 27017" dir=in action=allow
protocol=TCP localport=27017
Ok.
C:\>
```

This code says that all of the incoming traffic is allowed on port 27017, so any application servers can connect to the mongod.

Encrypting Data

You have seen that MongoDB stores all its data in a **data directory** that in Windows defaults to C:\data\db and /data/db in Linux. The files are stored unencrypted in the directory because there's **no provisioning of methods** for **automatically encrypting** the files in Mongo. Any attacker with **file system access** can read the data stored in the files. It's the application's responsibility to ensure that **sensitive information** is **encrypted** before it's written to the database.

Additionally, operating system-level mechanisms such as **file system-level encryption** and permissions should be implemented in order to prevent unauthorized access to the files.

Encrypting Communication

It's often a requirement that communication between the **mongod** and the **client** (mongo shell, for instance) is encrypted. In this setup, you will see how to add one **more level of security** to the above installation by configuring SSL, so that the communication between the **mongod and mongo shell** (client) happens using a SSL certificate and key.

It is recommended to use SSL for communication between the server and the client.

Starting from Version **3.0**, most of the MongoDB distributions now have **support included for SSL**. The below commands are executed on a Windows platform.

The first step is to **generate the .pem file** that will contain the **public key certificate and the private key**. MongoDB can use either a self-signed certificate or any valid certificate issued by a certificate authority.

In this book, you will use the following commands to generate a self-signed certificate and private key.

1. Install **OpenSSL** and **Microsoft Visual C++ 2008 redistributable** as per the MongoDB distribution and the Windows platform. In this book, you have installed the **64-bit version**.
2. **Run the following command** to create a public key certificate and a private key:

```
C:\> cd c:\OpenSSL-Win64\bin
C:\OpenSSL-Win64\bin>openssl
```

This opens the OpenSSL shell where you need to enter the following command:

```
OpenSSL>req -new -x509 -days 365 -nodes -out C:\practicalmongodb\
mongodb-cert.crt -keyout C:\practicalmongodb\mongodb-cert.key
```

The above step generates a certificate key named **mongodb-cert.key** and places it in the C:\practicalmongodb folder.

3. Next, you need to **concatenate** the certificate and the private key to the **.pem file**. In order to achieve this, run the following commands at the command prompt:

```
C:\> more C:\practicalmongodb\mongodb-cert.key > temp
C:\> copy \b temp C:\practicalmongodb\mongodb-cert.crt > C:\practicalmongodb\
mongodb.pem
```

Now you have a .pem file. Use the following runtime options when starting the mongod:

```
C:\> C:\practicalmongodb\bin\mongod -sslMode requireSSL --sslPEMKeyFile C:\practicalmongodb\
mongodb.pem
```

```
2015-07-03T03:45:33.248-0700 I CONTROL   Hotfix KB2731284 or later update is installed, no
need to zero-out data files
2015-07-03T02:54:30.630-0700 I JOURNAL   [initandlisten] journal dir=C:\data\db\journal
2015-07-03T02:54:30.670-0700 I CONTROL   [initandlisten] MongoDB starting : pid=2
816 port=27017 dbpath=C:\data\db\ 64-bit host=ANOC9
2015-07-03T02:54:30.670-0700 I CONTROL   [initandlisten] targetMinOS: Windows 7/Windows
Server 2008 R2
2015-07-03T02:54:30.670-0700 I CONTROL   [initandlisten] db version v3.0.4
2015-07-03T02:54:30.670-0700 I CONTROL   [initandlisten] OpenSSL version: OpenSSL1.0.1j-fips
19 Mar 2015
2015-07-03T02:54:30.670-0700 I CONTROL   [initandlisten] build info: windows sys.
getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1')
BOOST_LIB_VERSION=1_49
2015-07-03T02:54:30.671-0700 I CONTROL   [initandlisten] allocator: system
2015-07-03T02:54:30.671-0700 I CONTROL   [initandlisten] options: { net: { ssl: {
  PEMKeyFile: "c:\practicalmongodb\mongodb.pem", mode: "requireSSL" } } }
2015-07-03T02:54:30.680-0700 I NETWORK   [initandlisten] waiting for connections
on port 27017 ssl
2015-07-03T03:33:43.708-0700 I NETWORK   [initandlisten] connection accepted from
127.0.0.1:49194 #2 (1 connection now open)
```

■ **Note** Using a self-signed certificate is not recommended in a production environment unless it's a trusted network because it will leave you vulnerable to man-in-the-middle attacks.

You will next connect to the above mongod using the mongo shell. When you `run mongo` with a `-ssl` option, you need to either specify `-sslAllowInvalidCertificates` or `-sslCAFile`. Let's use `-sslAllowInvalidCertificates`.

Open a terminal window and enter the following:

```
C:\>C:\practicalmongodb\bin>mongo --ssl --sslAllowInvalidCertificates
2015-07-03T02:30:10.774-0700 I CONTROL Hotfix KB2731284 or later update is installed, no
need to zero-out data files
MongoDB shell version: 3.0.4
connecting to: test
```

Provisioning Using MongoDB Cloud Manager

In the starting of the chapter, you learned how to `install` and `configure MongoDB` using Windows and Linux. In this part of the chapter, you will look at how to use `MongoDB Cloud Manager`.

Mongo DBCloud Manager is a `monitoring solution` built in by the developer of the database. Prior to version 2.6, MongoDB Cloud Manager (formerly known as MongoDB Monitoring Service or `MMS`) was used for `monitoring and administering MongoDB only`. Starting from version 2.6, major enhancements have been introduced to MongoDB Cloud Manager including `backup, point-in-time recovery, and an automation feature`, making the task of operating MongoDB `simpler` than before. The automation feature provides power capabilities to administrators to `quickly create, upgrade, scale, or shut down MongoDB instances` in few clicks.


In this part of the book, you will see how to get started with MongoDB Cloud Manager. You will deploy a standalone MongoDB instance on AWS using MongoDB Cloud Manager.

When you start with MongoDB Cloud Manager, it asks to install an automation agent on each server, which is then used by the MongoDB Cloud Manager for communicating with the server.

In order to start provisioning, you first need to create your profile on `MongoDB Cloud Manager`.

Enter the following URL: <https://cloud.mongodb.com>. Click the Login or Sign up for Free button, based on whether you have an account or not.

Since you are starting for the first time, clicked the `Sign up for Free` button. This sends you to the page depicted in Figure 5-1.

 Account Profile

First Name

ABC

Last Name

DEF

Email Address

abc@gmail.com

Password

.....

Your password must be at least (8) characters long and contain at least one letter, one digit and one special character.

Job Function

Other

What is your experience level with MongoDB?

About to Enter Production

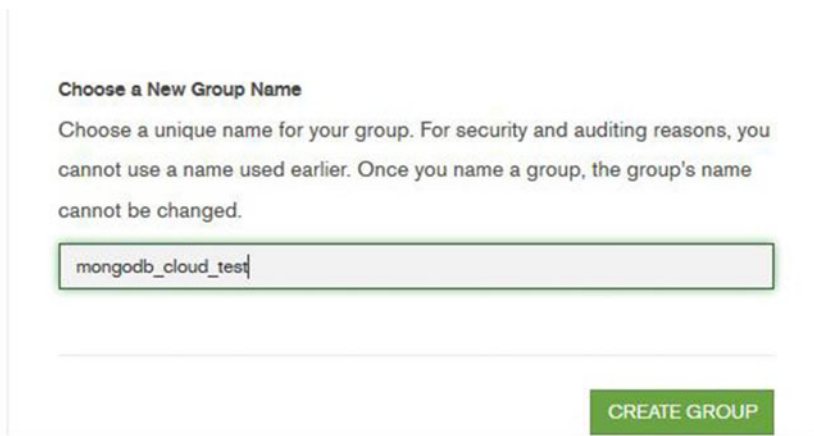
Joining an existing Cloud Manager Group at your organization?

CONTINUE

Figure 5-1. Account Profile

You will be creating a new profile. However, MongoDB provides an option for joining as existing **Cloud Manager** group.

Enter all the relevant details, as shown in Figure 5-1, and click Continue. This sends you to the page for providing company information. Once you complete the profile and company information, accept the terms and click the Create Account button. This completes the profile creation. The next step is to create a group (Figure 5-2).



Choose a New Group Name

Choose a unique name for your group. For security and auditing reasons, you cannot use a name used earlier. Once you name a group, the group's name cannot be changed.

mongodb_cloud_test

CREATE GROUP

Figure 5-2. Create Group

Provide a unique name for the group, and click Create Group. Next is the deployment selection page shown in Figure 5-3, where you have the option to build a new deployment or manage an existing deployment.



You haven't set anything up yet -- let's get started!

Build New MongoDB Deployment

OR

Manage Existing MongoDB Deployment

Will someone else be setting up your MongoDB deployment? [Add them to your group.](#)

Figure 5-3. Deployment

Select to build a **new deployment**. Next, you'll be prompted for the location of where to build the deployment (i.e. **Local**, **AWS**, or other **remote environment**). In this example, select AWS. Clicking the Deploy in AWS option leads you to choose between provision on your own and using Cloud Manager to provision.

Select the **"I will Provision"** option, which means you will be using a machine that is already provisioned to you on AWS.

The next screen provides options for the **deployment type** (i.e. standalone, replica set, or sharded cluster). You are doing a **standalone deployment**, so click the Create Standalone box. This sends you to the screen shown in Figure 5-4.

• Provide details for your standalone instance

Instance Name

Give your instance a name:

test

Data Directory Prefix

Your data will go in this directory on your servers.

If you are planning to deploy to servers running Windows, it is highly recommended that you change this to a Windows-style path such as C:\MMSAutomation\data.

/data

GO BACK

CONTINUE

Figure 5-4. Details for a standalone instance

Provide the instance **name** and **data directory prefix**, and click Continue. Next is the screen shown in Figure 5-5, which prompts you to **install an automation agent** on each server

Install an Automation Agent on each server.

Before we can create your MongoDB deployment, you'll need to download and follow the instructions for installing an Automation Agent on each server.

Please ensure that ports 27000 through 27020 are not currently in use by other processes on these servers, and that your firewall allows traffic on these ports between the servers.

I have 1 servers:

Your Server

INSTALL AGENT

Select your server's OS:

RHEL/CentOS 7X - RPM

RHEL/CentOS (5.X, 6.X)/SUSE/Amazon Linux - RPM

Ubuntu (12.04+) - DEB

RHEL/CentOS 7X - TAR

Other Linux - TAR

Mac OS X - TAR

Windows - MSI

CONTINUE

Install 1 more agent(s)

Figure 5-5. Installing an automation agent

50

www.it-ebooks.info

This screen has an option for specifying the number of servers. In this example, you specify 1. Next, you need to specify the platform. Choose Ubuntu. Then the screen in Figure 5-6 appears.

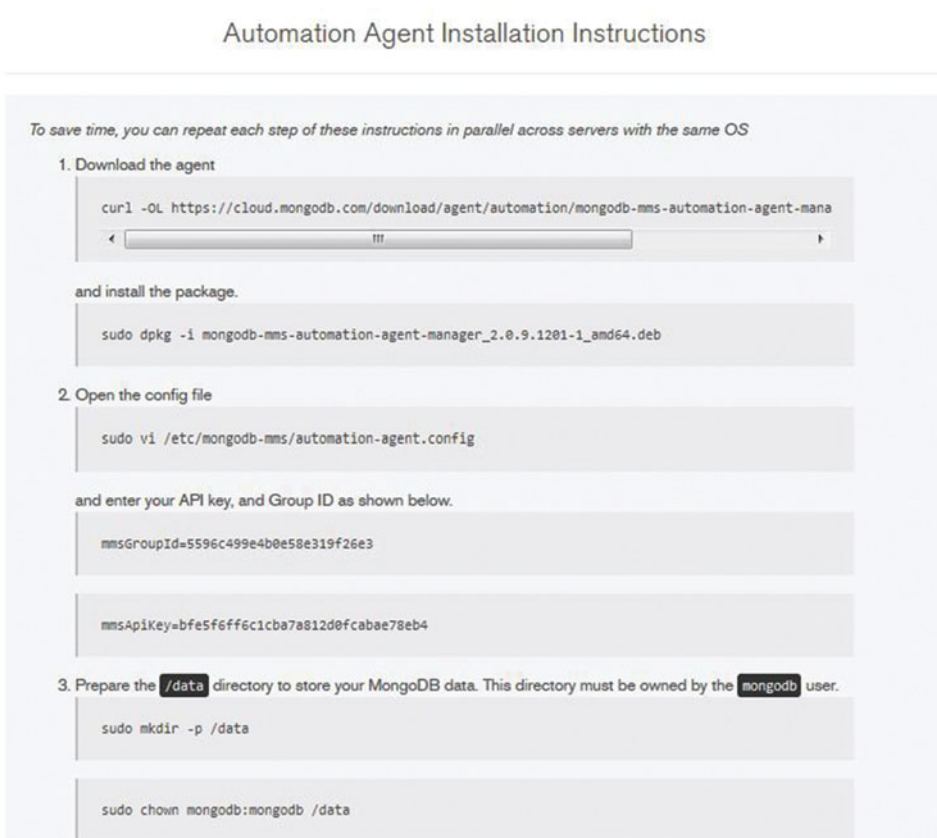


Figure 5-6. Automation agent installation instructions

Follow the steps.

Before you implement the step where you start the agent, you need to ensure that all the relevant ports are open (443, 4949, 27000 to 27018).

Once all the steps are completed, click the Verify Agent button. Post verification, if everything is working as needed, you'll see a Continue button.

When you click Continue, you will go to the Review and Deploy page shown in Figure 5-7 where you can see all of the processes that are going to get deployed. Here an automation agent downloads and installs the monitoring and backup agent.

Review & Deploy

You are about to deploy the following MongoDB processes on your servers. The Automation Agent will also install the other agents needed for monitoring and (optionally) backing up your deployment.

mongotest

State

Port

Version

Automation Agent

Monitoring Agent

Backup Agent

standalone

27000

3.0.4

GO BACK

DEPLOY

Figure 5-7. Review and deploy

Clicking the Deploy button takes you to the deployment page with the deploying changes status as “In progress.” When the installation is complete, the deployment status will change to “Goal State” and the provisioned server will appear in the topology view.

If your deployment supports SSL or using any authentication mechanism, you need to download and install a monitoring agent manually.

In order to vet whether all the agents are working properly or not, you can click the Administration tab on the console.

The Cloud Manager can deploy MongoDB replica sets, sharded clusters, and standalones on any Internet-connected server. The servers need only be able to make outbound TCP connections to the Cloud Manager.

Summary

In this chapter, you learned how to install MongoDB on the Windows and Linux platforms. You also looked at some important configurations that are necessary to ensure secure and safe usage of the database. You concluded the chapter by provisioning using MongoDB Cloud Manager.

In the following chapter, you will get started with MongoDB Shell.